HyperLIC

Xiaoqiang Zheng and Alex Pang
Computer Science Department
University of California, Santa Cruz, CA 95064
zhengxq@cse.ucsc.edu, pang@cse.ucsc.edu

ABSTRACT

We introduce a new method for visualizing symmetric tensor fields. The technique produces images and animations reminiscent of line integral convolution (LIC). The technique is also slightly related to hyperstreamlines in that it is used to visualize tensor fields. However, the similarity ends there. HyperLIC uses a multi-pass approach to show the anisotropic properties in a 2D or 3D tensor field. We demonstrate this technique using data sets from computational fluid dynamics as well as diffusion-tensor MRI.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques;

Keywords: hyperstreamlines, LIC, symmetric tensors, anisotropy, animation, direct volume rendering

1 INTRODUCTION

Tensor data is useful in many medical, mechanical and physical applications. Each tensor is a 3x3 matrix that contains nine unique quantities. To help comprehend such a large volume of information remains a difficult challenge for visualization research.

In this paper, we introduce a new method for visualizing symmetric tensor fields. It works on both 2D and 3D fields, as well as 2D manifolds in 3D. The visualizations are specially good at showing the anisotropy in the tensor fields and the resulting images resemble those of LIC. Similar to LIC, a white noise texture is needed. Conceptually, every pixel in the resulting visualization is calculated using an area (or volume, for 3D) averaged noise texture. The shape of the area (or volume) is determined by the local tensor field. Carrying this process out on a point by point basis will result in a blurred image as this is akin to low pass filtering. A better strategy is to think of the process as placing primitives such as squares or circles in 2D, cubes or spheres in 3D, along the path of a hyperstreamline. These primitives are deformed by the tensor field, and the resulting swept area or volume identifies the parts of the noise texture that will contribute to the intensity of a pixel. Now, the coherence of sampled noise textures start to show up.

What we are seeing in the HyperLIC visualizations are the anisotropy in the tensor field. Anisotropy in 3D tensor fields can be classified into three types: (a) linear or highly anisotropic characterized by the dominance of one eigenvalue, (b) planar characterized by one very small eigenvalue, and (c) spherical or isotropic characterized by three roughly equal eigenvalues. HyperLIC is particularly good at distinguishing between linear and spherical tensor regions.

The rest of this paper is organized as follows: Section 2 reviews related visualization techniques; Section 3 describes the components of the HyperLIC algorithm, first for 2D tensor fields, then for 3D; Section 4 discusses some implementation issues; Section 5 illustrates the methods and describes the results; and Section 6 summarizes our findings.

Images and animations can be accessed online at: www.cse.ucsc.edu/research/avis/hyperlic.html.

2 RELATED WORK

We review several classic and recent works that are related to this paper in one way or another. These are: hyperstreamlines [5], adaptive filtering of noise fields [8], oriented tensor reconstruction [1], direct volume rendering of diffusion tensors [6], and line integral convolution [4].

Hyperstreamlines were introduced by Delmarcelle and Hesselink in 1993 [5]. A tensor field is first decomposed into three eigenvector fields. Hyperstreamlines are essentially streamlines constructed from one of the eigenvector fields. The other two eigenvector fields are then encoded as the change in the cross section along the streamline. From any seed point, three hyperstreamlines can be generated using one of the three eigenvector fields for the streamlines and the other two for the cross section. The technique does not provide a global view and users need to mentally fill in what is happening with the tensor field even in the vicinity of the seed point. Like streamlines, one cannot seed too many hyperstreamlines as clutter becomes an issue. For non-symmetric tensors, the rotational component are encoded as “wings” along the main hyperstreamlines. HyperLIC is similar to how hyperstreamlines handle the symmetric portion of the tensor field in the following manner – the volume swept out by a hyperstreamline roughly corresponds to the volume of the noise texture used by HyperLIC to calculate the intensity value at the seed point.

This brings us to the LIC algorithm for visualizing vector fields, introduced by Cabral and Leedom also in 1993. Given an input vector field and a noise texture with the same dimensions, a LIC image is generated by calculating a streamline at each point, and then calculating a weighted average of the noise textures along the streamline, to produce the intensity value at the seed point. HyperLIC is similar to LIC in that it calculates a weighted average of noise texture values along a streamline. However, instead of simply using noise texture values along the streamline, we use noise texture values in the vicinity of the streamline as well. This local vicinity is defined by how tensors along the streamline deform the space around it. Unlike hyperstreamlines, there is minimal observable difference when we switch the order of the eigenvector field used to generate the streamline (see Figure 4).

More recently, Sigfridsson et al. [8] introduced an algorithm for visualizing symmetric tensor fields by iteratively applying an adaptive filter to directionally smear a noise texture in the frequency domain. Several discrete predefined directional filters are employed to categorize the continuous tensor anisotropy. One disadvantage of this method is the discretization of the orientation of the filters. If an anisotropic linear tensor falls in between a pair of oriented filters, it can only be described by their joint effects. Because of this, some

contrast is lost in the result.

Recently, attention has focused on how to visualize diffusion tensor MRI images. One of the challenges is how to deal with the noisy nature of the tensor field particularly when tracing neural pathways [1]. Tracing essentially involves integrating a streamline using the principal eigenvector. In this method, moving least square regularization successfully overcame the noise problem and relatively coarse grid used in the data set. Similar to hyperstreamlines, seeding is an issue that needs to be addressed in a more general fashion. Since the neural pathway tracing were to highlight regions of high anisotropy, seeds were naturally initiated where the linear tensors were high. Like hyperstreamlines, the method provides crisp visualization of streamlines along an eigenvector, but does not provide a continuous, global view of the tensor field.

One approach that does attempt to provide a continuous global view is the adaptation of direct volume rendering to tensor fields presented by Kindlmann et al. [6]. Like [1], the tensor field is first analyzed with respect to its anisotropy and classified into three continuous categories: linear (anisotropic), planar, and spherical (isotropic) tensors. This property of the tensor field is then used as barycentric coordinates of a triangular transfer function that highlights regions of different anisotropic properties. Further enhancements are then provided using lit-tensors mixed with opacity gradient shading and hue-balls combined with deflection mapping [7]. The deflection mapping strategy seem to provide the most dramatic results, but would also depend on the granularity of the textures used.

This short list is by no means a comprehensive list of tensor visualization techniques. We have been working on deformation based approach to visualize tensor fields [3, 10, 11]. In [3], idealized objects such as lines, planes, and sub-volumes are deformed by a tensor field and visualized by the users as they interactively modify an interrogation vector. The main drawback of this approach is that users can only see the information in one direction at a time. This was further improved in [10] in two ways: (a) allow the normal vectors to change dynamically as the object is being deformed, and (b) allow entire volumes to be deformed by solving a system of linear equations that minimize the energy across a spring system connecting all the cells in the tensor volume. However, one drawback that persisted is that the techniques applies only to symmetric tensor fields. This limitation is removed in [11] as the tensor field is used to affect how synthetic light would interact with the tensor volume. Three methods are described: (a) Light rays are shot from a certain direction into the tensor volume. These rays are influenced by the surrounding tensor field and bent as they traverse through the volume. The tensor is visualized by both the nature of the bent rays which shows paths of greatest influence by the tensor field. (b) Instead of looking at the bent rays, we observe how the rays are deposited on a receiving plate. This is similar to caustic effects from photon maps, but shows the convergence or divergence of the rays through the tensor volume. (c) We also use the concept of treating the tensor volume as a special lens that distorts an image. Using backward ray tracing through the tensor volume, we generate image distortions that also show internal properties of the tensor field. HyperLIC can also be considered a deformation approach in the sense that the tensor field is used to deform (and thereby define) the space of noise texture that is used in the calculation for the intensity at a point.

3 METHODS

In this section, we first discuss the development of 2D HyperLIC to visualize tensor fields, then later discuss how it can be extended to 3D. A 2D symmetric tensor can be uniquely determined by its magnitude, anisotropy and direction. HyperLIC displays the anisotropies and directions of the tensor fields through the intensities of the synthetic texture, while mapping the magnitude to color.

We achieve this goal by locally filtering an input white noise image. The anisotropies of tensors are represented by the sharpness of the output image. A highly anisotropic tensor, such as a linear tensor, can be reduced to a vector. HyperLIC shows them in LIC-like styles with dense and sharp lines by averaging the input image along the streamline. An isotropic tensor, such as an identity tensor, shows no directional preference, because all vector will be transformed into themselves. HyperLIC features them in a smoothed-out style by averaging in a local area. General tensors are all combinations of these two extreme cases, so we illustrate them as combinations of the two different styles in the output image by averaging the input image in an area locally defined by the tensors.

We also discuss the animation in HyperLIC. In linear regions, the animation should look LIC-like style, while in isotropic regions, the motion should appear directionless.

Extending the algorithm to 3D is a natural extension from 2D. However, like other attempts of extending LIC to 3D, the challenge is how to deal with heavy visual clutter. Fortunately, for tensor fields, it makes sense to encode the anisotropy to transparency thereby allowing one to study the spatial structure of the data. We demonstrate results of this method using diffusion tensor MRI brain data.

To improve the rendering of the volume texture, a good shading method can successfully enhance the depth perception. However, the raw data are too noisy to generate accurate gradients for use as normals in the shading model, so we employ a filtering algorithm to smooth out the noise for the generation of the smooth gradients.

Before we start describing HyperLIC, we introduce some preliminary tensor transformations on the data.

3.1 Tensor Processing

Tensors exist in 3D physical space. Before we apply HyperLIC, we need to transform them to computational space where the texture is defined. If it is 2D HyperLIC, we need to project the 3D tensors to obtain 2D tensors. We also discuss a method of remapping the tensors to highlight the desired features before we pass them to HyperLIC.

3.1.1 Tensors in Curvilinear Grids

It is quite common for computational fluid dynamics applications to distinguish between computational space and physical space. For example, if we want to study the tensors on a wing geometry this would correspond to a slice in computational space. Alternatively, one may want to study the tensors on a slice through physical space. In either case, 3D tensors need to be projected onto a surface. This surface can be expressed as $S(u, v)$, where u and v are computational coordinates. We also need to find a transformation between tensors in physical and computational spaces. Let the three principal axes be: $Q_1 = \frac{\partial S}{\partial u}$, $Q_2 = \frac{\partial S}{\partial v}$ and $Q_3 = Q1 \times Q2$. Also define:

$$W = \begin{pmatrix} Q_{1x} & Q_{2x} & Q_{3x} \\ Q_{1y} & Q_{2y} & Q_{3y} \\ Q_{1z} & Q_{2z} & Q_{3z} \end{pmatrix} \quad (1)$$

Assume the tensor is represented as P in physical space and represented as C in computational space. The general matrix transformation between the two coordinate system, $C = W^{-1} \cdot P \cdot W$, does not apply to our application. We want both P and C the power to transform a unit sphere in their own space, S_p and S_c respectively, into the same ellipsoid in physical space. Because S_p and S_c are

both unit sphere, $S_p = R \cdot S_c$, where R is a unitary matrix. This can be expressed as:

$$P \cdot S_p = P \cdot R \cdot S_c = W \cdot C \cdot S_c \tag{2}$$

This can be reduced to:

$$C = W^{-1} \cdot P \cdot R = POLAR_S(W^{-1} \cdot P) \tag{3}$$

where $POLAR_S(X)$ is the symmetric part of X in a polar decomposition. This transformation guarantees that C in computational space has the same effect as P in physical space.

The justification for the transformation above is because a matrix M can always be represented as the product of a symmetric matrix T_s and a unitary matrix R such that $R^T \cdot R = I$.

$$M = T_s \cdot R \tag{4}$$

3.1.2 Tensor Projection

Once the tensor is in the computational space, we need to project it onto the surface, $S(u, v)$. The first two axes, Q_1 and Q_2 are on the surface while Q_3 is perpendicular to this surface. So we discard all the components associated with Q_3, resulting in:

$$C_2 = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \tag{5}$$

where C_2 is now a 2×2 tensor. Through this projection, a planar tensor along the surface is expressed as a 2D tensor without loss of information.

3.1.3 Tensor Remapping

Tensor remapping allows us to vary the contrast level of the resulting image. Given the users specified parameter n, the remapped tensor is:

$$T_r = T^n \tag{6}$$

where T and T_r are the original and remapped tensors respectively. The exponentiation operation is that defined on matrices. For a general tensor T with eigenvector matrix E and diagonal eigenvalue matrix $diag(\lambda_1, \lambda_2, \lambda_3)$, the above equation can also be written as:

$$T_r = E \cdot diag(\lambda_1^n, \lambda_2^n, \lambda_3^n) \cdot E^{-1} \tag{7}$$

Figure 6 illustrates the effect of varying the parameter n.

3.2 Basic Idea

Prior to visualizing the symmetric tensor fields, they are first decomposed into orthogonal eigenvector fields: $E = \{e_1, e_2, e_3\}$. The corresponding eigenvalues are: $\Lambda = \{\lambda_1, \lambda_2, \lambda_2\}$.

The basic idea of the algorithm is as follows: given a 2D symmetric tensor field and an input noise field, a geometric primitive is placed over each tensor. This primitive is going to be deformed by the tensor field. Using the deformed primitive at each location, The noise texture values under each deformed primitive are averaged together to give the pixel value of the resulting image. This procedure is similar to the DDA algorithm described in [4] where the noise texture under each vector line glyph is used to calculate pixel intensities. When this process is carried out over tensor fields (as opposed to vector fields), the anisotropic properties of the underlying tensor field is revealed.

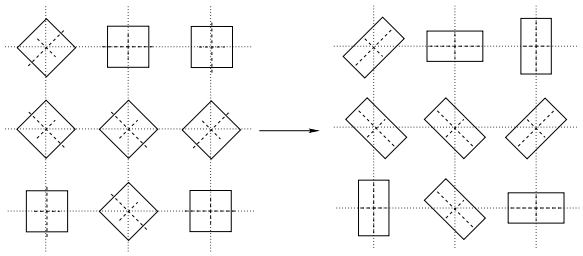


Figure 1: Squares primitives aligned with eigenvectors.

We experimented with two types of geometric primitives. The first type is a circle. In general, the circle is deformed into an ellipse, which is widely considered as an elegant glyph for a symmetric tensor. An advantage of this type of primitive is we do not even need to decompose the tensor fields into eigenvector fields. Circles work well with isotropic tensors, but produces undesirable artifacts in regions with linear tensors.

The second type of primitive is a square, oriented over each tensor such that the sides are aligned with the orthogonal eigenvectors – hence the need to first find the eigenvectors. The tensor deformation of these eigenvector aligned squares results in rectangles as shown in Figure 1. The scaling factors are simply the eigenvalues. In regions with purely linear tensors, the square primitives are reduced to line segments and hence produce results similar to the DDA algorithm. In regions with isotropic tensors, HyperLIC is reduced to a smoothing algorithm, which averages the input texture within a local region. It produces a blurred image without sense of directions. In summary, HyperLIC maps the anisotropic properties of tensors to sharp and blurred regions in the resulting image.

3.3 Conceptual Algorithm

In LIC, the DDA is carried one step further, where the noise texture is integrated over the streamline rather than just a straight line segment [4]. This allows one to capture more subtle features and improve contrast. This enhancement can be carried out with HyperLIC in the following fashion. One of the eigenvector fields is selected to generate a hyperstreamline using N integration steps. This hyperstreamline acts as a skeleton of the texture sampling region. Next, we draw hyperstreamlines using the other eigenvector at each of the N points along the first hyperstreamline. As shown in Figure 2, these two steps creates a sampling strip along the first hyperstreamline.

Where the tensors are highly linear, the sampling strip is essentially the primary hyperstreamline and HyperLIC reduces to LIC. With isotropic tensors, this sampling strip forms a square which acts a smoothing filter on the input image. This improved algorithm works better than its basic version just as LIC works better than DDA version.

3.4 Multi-pass Approach

While the conceptual algorithm captures subtle details of the underlying tensor fields it is also much expensive than LIC. This is because for each pixel, HyperLIC needs to create a 2D sampling area as opposed to 1D in LIC. Here, we describe a multiple-pass approach that accelerates the computation of HyperLIC.

Let P be a point in the tensor field, I_n be the input noise texture image, and I_o be the output HyperLIC image. Then, an output pixel

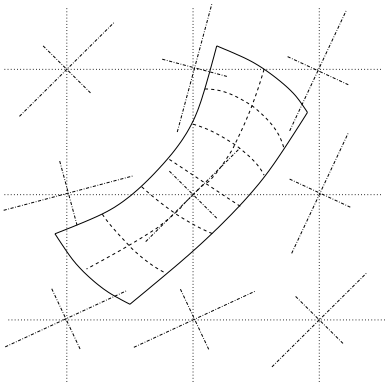


Figure 2: Sampling strip defined by a primary hyperstreamline and a set of orthogonal hyperstreamlines.

is defined as:

$$I_o(P) = \frac{\sum_{i=-N}^{N}\sum_{j=-N}^{N} k(i,j) I_n(P_{i,j})}{\sum_{i=-N}^{N}\sum_{j=-N}^{N} k(i,j)} \quad (8)$$

$$P_{i,j} = P_{i,j-1} + \lambda_1(P_{i,j-1}) e_1(P_{i,j-1}) \Delta t \quad (9)$$

$$P_{i,0} = P_{i-1,0} + \lambda_2(P_{i-1,0}) e_2(P_{i-1,0}) \Delta t \quad (10)$$

$$P_{0,0} = P \quad (11)$$

where $\lambda_l(X),\ e_l(X),\ k(i,j),\ l = 1,2$ are the lth eigenvalues, eigenvectors and the weight function at point X. Δt is the integration time step. In our experiment, the joint kernel function $k(i,j)$ is defined as the product of two kernel functions $k_1(i)$ and $k_2(j)$. k_1 and k_2 are constant for static images, and are defined in Equation 19 for animation sequences.

$$k(i,j) = k_1(i) k_2(j) \quad (12)$$

If we define $I_1(P)$ as:

$$I_1(P) = \frac{\sum_{j=-N}^{N} k_2(j) I_n(P_j)}{\sum_{j=-N}^{N} k_2(j)} \quad (13)$$

$$P_j = P_{j-1} + \lambda_2(P_{j-1}) e_2(P_{j-1}) \Delta t \quad (14)$$

$$P_0 = P \quad (15)$$

then, $I_o(P)$ can also be expressed as:

$$I_o(P) = \frac{\sum_{i=-N}^{N} k_1(i) I_1(P_i)}{\sum_{i=-N}^{N} k_1(i)} \quad (16)$$

$$P_i = P_{i-1} + \lambda_1(P_{i-1}) e_1(P_{i-1}) \Delta t \quad (17)$$

$$P_0 = P \quad (18)$$

Equations 13 and 16 are essentially the output images of the unnormalized LIC on $\lambda_2 e_2$ and $\lambda_1 e_1$ vector fields with input images I_n and I_1 respectively. Thus, we can reduce HyperLIC to a multiple-pass unnormalized LIC. In the first pass, we apply the unnormalized LIC on $\lambda_2 e_2$ using the input image I_n and kernel k_2 to get an intermediate image I_1. In the second pass, we use the output image from the previous pass, I_1, as the input image and apply unnormalized LIC on $\lambda_1 e_1$ with kernel k_1 to generate the output image I_o.

Such a two-pass approach greatly reduces the amount of redundant computation in the conceptual algorithm. In each pass, only a 1D sampling is needed. To further improve performance, Fast LIC [9] is implemented in each pass such that the convolutions for each pixel is performed incrementally in order to reduce the overall number of streamline calculations. This allows one to interactively explore 2D tensor fields using HyperLIC.

In the vicinity of linear tensors, the first pass filters the noise texture into a LIC-like image with dense and sharp lines. In the second pass, the intermediate image remains unchanged if the orthogonal eigenvalues are too small, implying a region of high anisotropy. If the eigenvalues are very high, which means low anisotropy, it blurs out the image across the line direction. It achieves the same goal as the conceptual algorithm in a different but much faster way.

We note that the conceptual algorithm and the multiple-pass improvement both depend on which eigenvector is processed first. Theoretically, the order in which the eigenvector fields are processed will affect the final image. In practice, the differences are not noticeable (see Figure 4).

The differences are due to the spatial variation in the tensors. That is, following the major eigenvector for X steps from P and then switching to the minor eigenvector for Y steps may end up in a different location than following the minor for Y steps and the switching to the major for X steps. In symmetric tensors, the error is on the order of $O(L)$. In our experiments, L is set to $1/25$ of the size of images. Since L is relative small for each point, the difference is negligible both theoretically and empirically. It means we can interpret the results from HyperLIC as only affected by the underlying tensor field and not by the order of how the eigenvectors were processed.

Although the ordering of eigenvectors is unimportant, how the eigenvectors are classified dramatically affects the quality of output images. That is to say, eigenvectors must be classified into major or minor fields. This algorithm produces nice results in most regions. However, because the major and minor eigenvectors may switch near critical points in the tensor field, integrating along an eigenvector field generates sudden transitions on an otherwise smooth tensor.

To avoid this problem, we label the eigenvector fields as follows: First, we find a point with high anisotropy and label its eigenvectors as 1st and 2nd. Next, we iteratively label its neighbors consistently until all the points are labeled. An important point is that we set the eigenvectors as unchanged through the critical points. This labeling algorithm produces smoothly changing eigenvectors. For a smooth tensor field, there is no sudden change of major or minor eigenvectors.

3.5 Sign Indeterminacy

HyperLIC images show the orientation of anisotropy but it doesn't show the direction of the major eigenvector. This is because the sign of the eigenvector is indeterminate. However, we can impose a consistent direction on the eigenvectors by using the signs of the eigenvalues. The idea is inspired by the behavior of charged molecules. Placing a positively charged molecules M in an electric field dispels all other positively charged molecules and attracts all other negatively charged molecules. In another word, we can understand the sign of each molecules in this field by observing their motion relative to M.

To generalize this idea, we first synthesize a simple interrogational vector field. We then make the eigenvector directions follow this synthetic vector field. The synthetic vector field we chose in our experiment is $v(x,P) = x - P$, where P is a user-specified attracting point and x is a location in the vector field. The sign of an eigenvector e_i is chosen to make $\lambda_i e_i \cdot v(x,P) > 0$. With this direction-deciding algorithm, all eigenvectors with positive eigen-

values flow away from the attracting point while all eigenvectors with negative eigenvalues flow to the attracting point. We can easily identify the signs of the eigenvalues by observing the flow direction toward or away from the attracting point.

In our experiments, a single attracting point is enough. However, for more complicated data, we may need multiple attracting points or a more complex interrogational vector field.

3.6 Animation

After deciding the directions of eigenvectors, we propose an animation technique similar to that introduced by Cabral for LIC [4] by varying the kernel functions according to time T.

$$k_l(w,T) = \frac{1+\cos(cw)}{2} \times \frac{1+\cos(dw+\beta T)}{2}, l = (1,2) \quad (19)$$

$$k(w_1,w_2,T) = k_1(w_1,T)k_2(w_2,T) \quad (20)$$

where c and d are two constants and β is the phase shift of the ripple function. $k_l(w), l = 1,2$ are kernel functions defined on each of the eigenvectors. The joint kernel function $k(w_1,w_2)$ is defined as the produce of $k_1(w_1)$ and $k_2(w_2)$.

This time varying kernel function constantly shifts toward the eigenvector directions. For a linear eigenvector, only the kernel function for the major eigenvector has any effect, so HyperLIC produces a LIC-like animation flow. For an isotropic eigenvector, the kernel function is a multiplication of two simultaneously shifting kernel functions, so the flow pattern appears confused and ambiguous.

3.7 Extending HyperLIC to 3D

The HyperLIC algorithm described thus far works well in 2D. It shows the anisotropic properties of the tensors as varying intensities in the output image and maps tensor magnitude to color. The signs of eigenvalues are mapped into the directions of eigenvectors and visualized through animations. Although all the information of 2D symmetric tensors are visualized, the extra information in the 3rd dimension are lost during tensor projection.

Fortunately, HyperLIC can be easily extended to 3D. The conceptual 3D HyperLIC works by averaging the input 3D image using a 3D sampling strip defined by the three eigenvectors. The two-pass 2D HyperLIC is extended to three passes in 3D. During each pass, one set of eigenvectors is used to apply unnormalized LIC on the input volume with the corresponding kernel functions, then pass the output volume as the input in the next pass. The final 3D texture are then rendered using direct volume rendering.

We use a 3D diffusion tensor MRI brain data to demonstrate 3D HyperLIC. In medical data, fiber traces are important features. Fibers are represented by tensors with high anisotropy. Tensors associated with white material are relatively isotropic. HyperLIC visualizes the fibers as sharp lines and the rest of the brain as blurred regions. The global structure is insensitive to noise in the underlying data because HyperLIC smears out the local noise. To highlight the fibrous regions, we chose to map the local variance in the output volume to transparency.

$$var(P) = \sqrt{\frac{\sum (I_o(P) - I_o(P_n))^2}{6}} \quad (21)$$

where P_n are P's neighbors. This form of variance is very close to the magnitude of local gradient. However, because the central difference form of gradient acts as a low pass filter on HyperLIC high frequency noise texture, we use Equation 21 to measure the sharpness in HyperLIC output image. Using a transfer function that maps this variance to transparency, 3D HyperLIC can extract the fibrous regions of the brain data set. Tensor magnitude are still mapped to color as before.

Accurate and smooth shading is crucial to visual perception. In traditional volume rendering algorithm, the normal of the surfaces are mapped to the gradient of data values. We find this to produce very noisy gradients. Instead, we apply Equation 21 repeatedly to obtain a smoother normal. After each step, the cells are assigned the average of the old values in its neighboring cells. The result is smoother shading on the fiber structures which greatly enhances the depth perception of brain. An important point to note is that smoothed variances are only used to calculate the gradient for shading purposes. The transparency is still mapped to the original variances. Hence, what we are seeing accurately reflects what is in the data.

4 IMPLEMENTATION ISSUES

4.1 Inverse HyperLIC algorithm

The default HyperLIC algorithm shows strong directional information about the major eigenvectors, but it does not produce a strong visual effect about the other components. To compensate, we can switch the tensors in the preprocessing stage – so the minor eigenvectors become the majors. As a result, the images strongly resemble the minor hyperstreamlines (see Figure 3(c)).

4.2 Storage Requirement

3D HyperLIC is computationally expensive and requires a lot of memory. For a high resolution data set, 512^3 volume of floats in our experiment, the input and output volumes take up to 800 Mb of memory. To handle this huge memory demand, we employ solve the problem one layer at a time.

HyperLIC is a locally based algorithm. A voxel in the output volume is only effected by voxels within a certain distance of the corresponding input voxel. Because the eigenvalues are normalized, the maximum radius of the effective area is $2N+1$ where N is the integration length. So, when we compute the output volume for layer Y, only layers from $Y-N$ to $Y+N$ in input volume are needed. We implemented this method with an output buffer of 1 layer, and an input buffer of up to $2N+1$ layers. During each step, we update the output layer with the input layers. At the end of each step, we write the output layer into a temporary file and update the next layer of the input buffer, then go to the next step.

The results of the output volume after each pass in HyperLIC is stored in a temporary file. After each pass, we use the output file in the previous pass as input in the next pass. We implement this algorithm in parallel, by computing each output layer separately. Because the input textures are read-only, they are shared by all the rendering threads. When a rendering thread for an output layer is started, only the input layer not already in memory is loaded. This strategy reduces the memory requirements and makes the implementation very amenable to parallel computation on machines with multiple processors.

4.3 Rendering

3D HyperLIC produces a colored volume. In our experiment, this volume is of 512^3. Interactive exploration of this large volume is a key component to understand the structure of the tensor data.

We employ a hardware-accelerated texture mapping algorithm to render this volume. Before the rendering, three sets of volume textures are generated, each of them are sliced along one of the three texture coordinates. During the rendering of a given viewpoint, we

pick the set of texture associated with the axis that best faces the camera. This is determined by computing the dot product of all three candidate axes and the camera and use the one with maximum absolute value.

This strategy works well in 3D, but it may produce undesirable artifacts in the animation when switching between two different sets of textures as the viewpoint changes.

5 RESULTS

In this section, we present results from HyperLIC on different data sets. The first data we experimented with is the single point load data. This stress tensor data is simple and throughly studied and therefore an excellent data for verifying the algorithm. The second data set is strain tensors derived from the flow past a cylinder with a hemispherical cap. This data is more complicated but it has also been used in other published tensor visualization methods. The third data set is diffusion tensor MRI of the brain. Neural pathways are represented by tensors with high anisotropies. We present results both in 2D projection and 3D for this data.

Figure 3 includes 2D HyperLIC results on single point load stress tensors from different view points. Figure 3(a) is a slice from the middle of the volume and viewed from the point load direction. It is mostly components from medium or minor eigenvectors. We see that the center of this slice is quite isotropic. Around the center is a ring formed by lines, which means tensors are highly anisotropic. It is the boundary where the minor eigenvalues are zero. From the animation available in the url, we see flow within the ring going towards the center, while flow outside the ring going away from the center. This reveals that stress inside the ring is stretching, outside the ring is compressing, and the ring itself is free of stress away or toward the center. A little further away from the ring, we find another isotropic area. After the yellow area, the tensors are mostly represented by dense and sharp lines oriented radially that shows high anisotropy in the regions.

Figures 3(b) and 3(c) are side views. Figure 3(b) contain mostly sharp lines, so eigenvalues are very large in these directions. An interesting feature appears near the surface shown in more detail in Figure 3(d). There we see a change of pattern. The sharp lines change directions rapidly in a very narrow strip. Further observation in animation reveals that flow near the surface is attracted to the point load, while flow further away is repelled. It clearly shows that the stress near the surface is stretching while compressing in the rest of the areas. It can be confirmed in the stress tensor equation and is hardly shown by other visualization methods. Figure 3(c) is the same view as (b), but shows inverse HyperLIC that highlights the minor eigenvectors.

Figure 5 are two results from strain tensors in flow past a cylinder with a hemispherical cap. Figure 5(a) is from the inner layer (closer to the geometry) and Figure 5(b) is from the middle layer (farther away from the geometry). From these two images, we can clearly see the tensor directions on the cap. Three blurred areas associated with isotropic tensors are clearly shown in Figure 5(b). We can also observe two degenerate wedge points. One is on the cylinder while the other is on the cap.

Figure 6 are from brain tensor data with different remapping parameters. Tensor remapping is used in the preprocessing stage of HyperLIC to vary the degree and contrast in which anisotropy is depicted. The higher the remapping parameter, the more linear HyperLIC looks. In Figure 6(a), most regions except a few are blurred. In Figure 6(b), more fibers are visible. We can observe the main fiber structure in this picture. This demonstrates that tensor remapping allows the user to vary the the level of detail in HyperLIC images.

Figure 7 shows results from the brain tensor data using 3D HyperLIC. Images are from different view points and rendered with shading. The light is always from the camera position. The main structures of the brain can be identified from these four views. Figure 8 illustrates the effect of rendering without and with shading. Finally, in Figure 8(c), we use chromadepth mapping [2] to further enhance depth perception with the aid of some inexpensive lens.

6 CONCLUSIONS

We presented a new technique for visualizing the anisotropy in symmetric 2D and 3D tensor fields. The technique provides a global, continuous representation of the field requiring minimal user input. Furthermore, the technique can produce animations that enhances perception and our understanding of the tensor field.

ACKNOWLEDGEMENTS

The brain dataset is courtesy of Gordon Kindlmann from the Scientific Computing and Imaging Institute, University of Utah, and Andrew Alexander from the W. M. Keck Laboratory for Functional Brain Imaging and Behavior, University of Wisconsin-Madison. This work is supported in part by NSF ACI-9908881 and NASA Cooperative Agreement NCC2-1260.

REFERENCES

[1] Leonid Zhukov an Alan Barr. Oriented tensor reconstruction: Tracing neural pathways from diffusion tensor MRI. In *Proceedings of Visualization 02*, pages 387–394, Boston, 2002.

[2] Michael Bailey and Dru Clark. Using ChromaDepth to obtain inexpensive single-image stereovision for scientific visualization. *Journal of Graphics Tools*, 3(3):1–9, 1998.

[3] E. Boring and A. Pang. Interactive deformations from tensor fields. In D. Ebert, H. Hagen, and H. Rushmeier, editors, *Proceedings IEEE Visualization '98*, pages 297–304. IEEE Computer Society Press, 1998. Tensor / Flow.

[4] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. *Computer Graphics Siggraph Proceedings*, pages 263–270, 1993.

[5] T. Delmarcelle and L. Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics and Applications*, 13(4):25–33, July 1993.

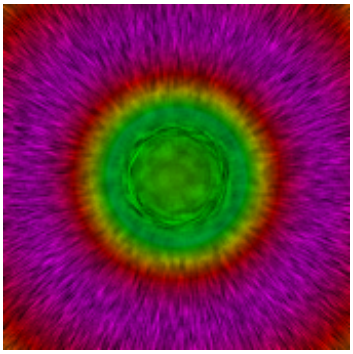
[6] G. Kindlmann, D. Weinstein, and D. Hart. Strategies for direct volume rendering of diffusion tensor fields. *Visualization and Computer Graphics*, 6(2), 2000.

[7] Gordon L. Kindlmann and David M. Weinstein. Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *IEEE Visualization*, pages 183–189, 1999.

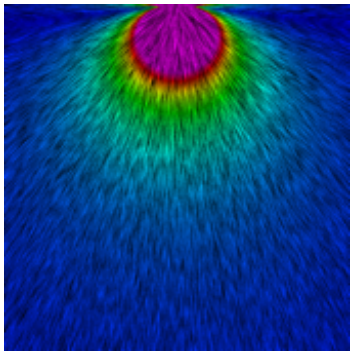
[8] Andreas Sigfridsson, Tino Ebbers, Einar Heiberg, and Lars Wigstrom. Tensor field visualization using adaptive filtering of noise fields combined with glyph rendering. In *Proceedings of Visualization 02*, pages 371–378, Boston, 2002.

[9] D. Stalling and H.-C. Hege. Fast and resolution independent line integral convolution. *Computer Graphics Siggraph Proceedings*, pages 249–256, 1995.

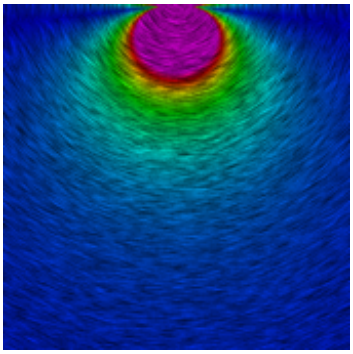
[10] Xiaoqiang Zheng and Alex Pang. Volume deformation for tensor visualization. In *Proceedings of Visualization 02*, pages 379–386, Boston, 2002.



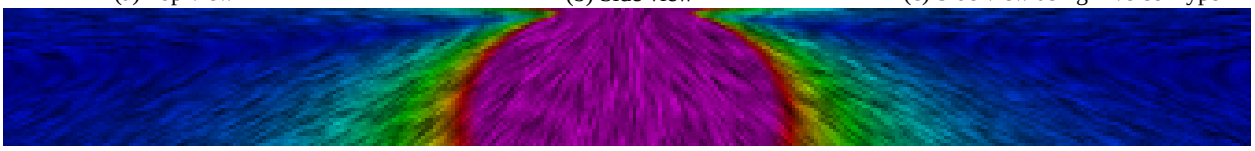
(a) Top view



(b) Side view

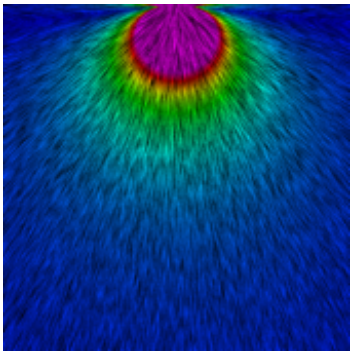


(c) Side view using Inverse HyperLIC

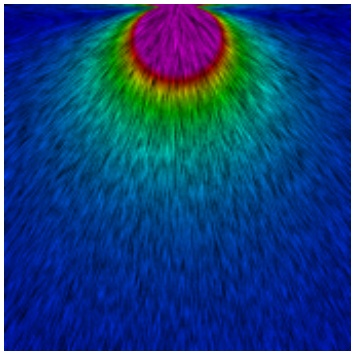


(d) Zoomed view of the top portion of (b)

Figure 3: Single point load data from different view points.

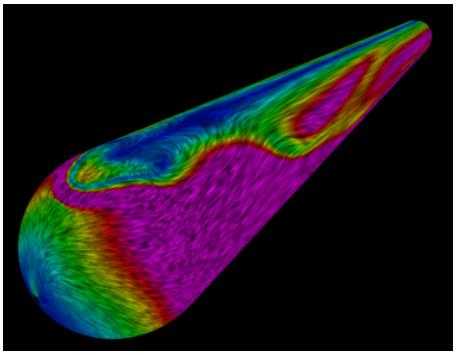


(a) Eigenvector order switched

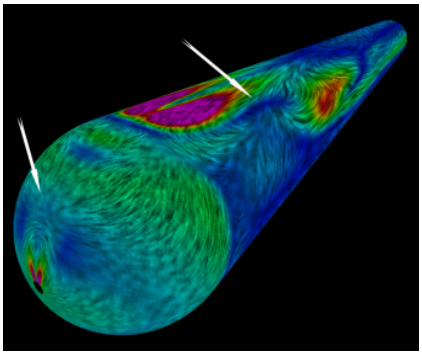


(b) Conceptual algorithm

Figure 4: There is minimal noticeable visual differences between the multi-pass algorithm (a) the minor eigenvector is processed first, and (b) the more expensive conceptual algorithm where sampling strips are calculated from hyperstreamlines.

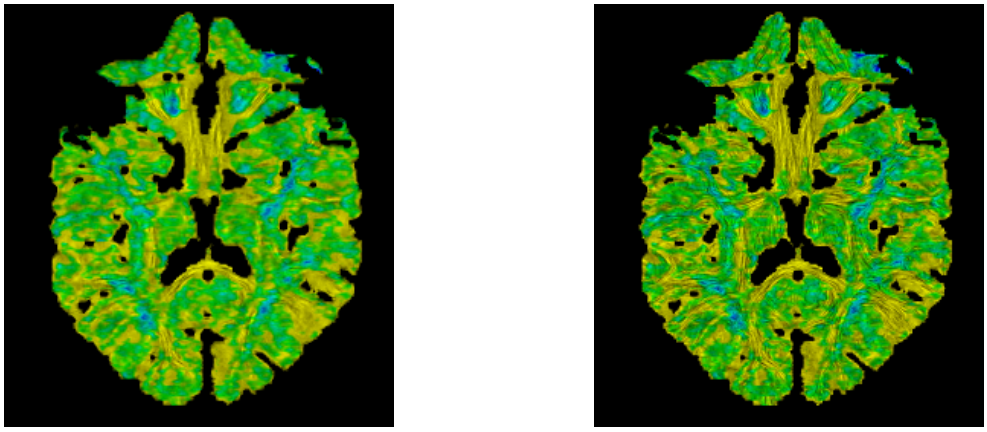


(a) Inner layer



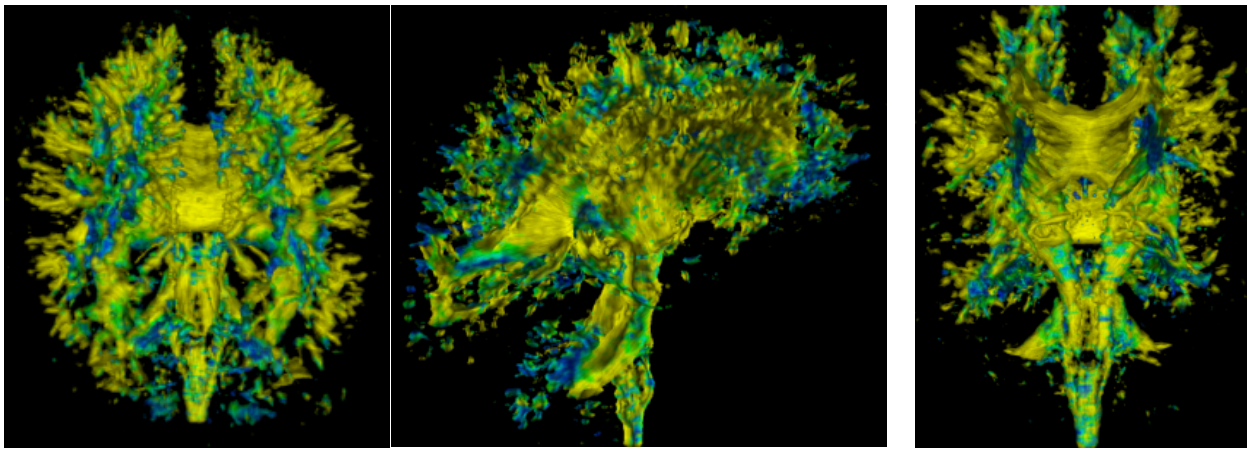
(b) Middle layer

Figure 5: Flow past a cylinder with hemispherical cap. HyperLIC of two different computational layers of the strain rate tensor. Arrows point to locations of degenerate wedge points.



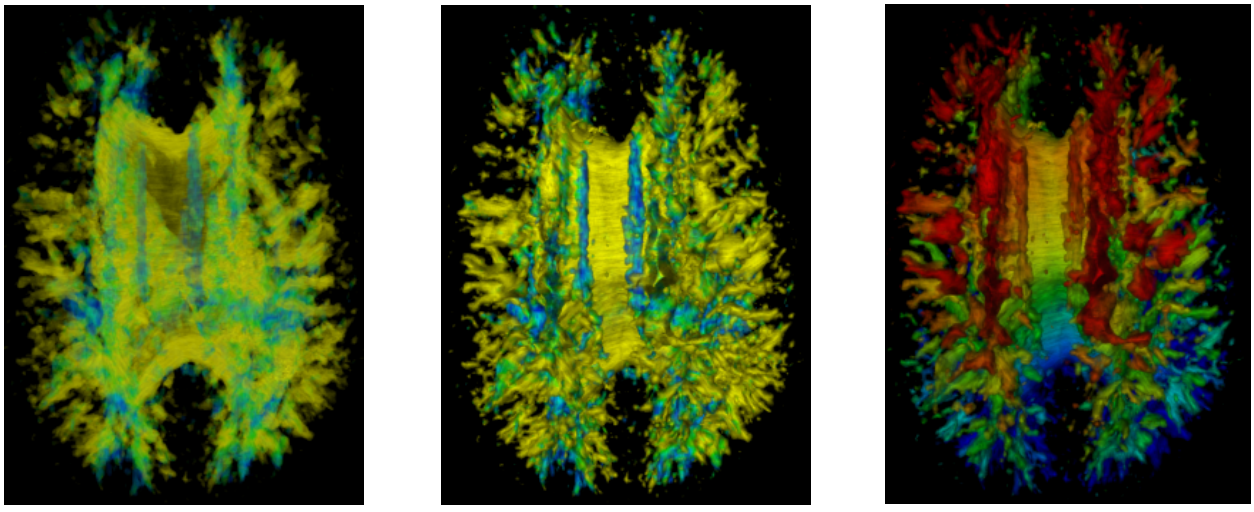
(a) Remapping with $n = 2$ (b) Remapping with $n = 8$

Figure 6: A 2D slice of the brain data using different tensor remapping parameters.



(a) Back view (b) Right view (c) Front view

Figure 7: 3D HyperLIC on medical brain tensor data from different view points.



(a) Unshaded (b) Shaded (c) Chromadepth

Figure 8: Comparison of different rendering technique

[11] Xiaoqiang Zheng and Alex Pang. Interaction of light and tensor fields. In *VisSym'03*, 2003. www.cse.ucsc.edu/research/avis/tensorray.html.